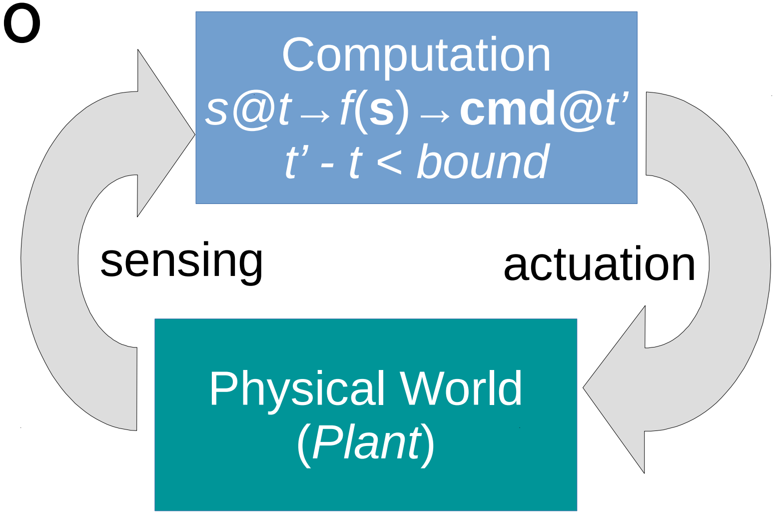
# IoT S&S:
# Real-Time

Gabe Parmer

# Real-Time/Cyber-Physical Systems

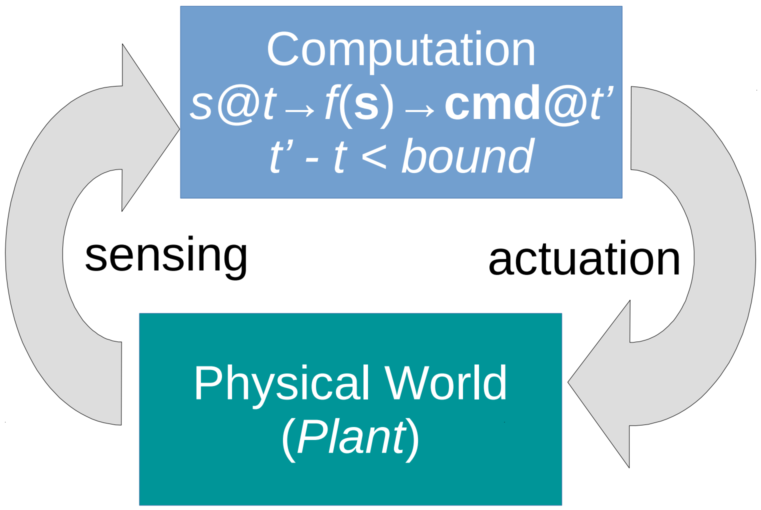
- Embedded systems + **deadlines**
  - The computation must adhere to the timing constraints of the physical world
  - Send command to actuator within a **bounded time** from receiving sensor information
  - We don’t write code thinking about time; *how do engineers program real-time systems???*



# Real-Time/Cyber-Physical Systems

Complexities

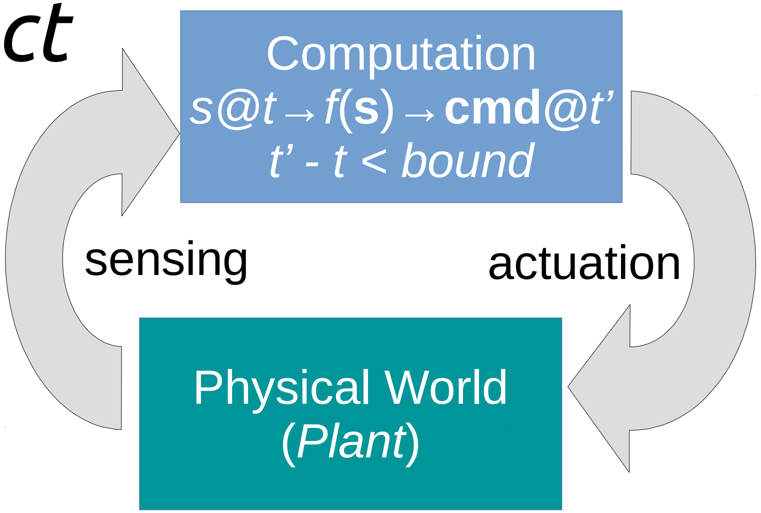
- *N* sensors, *M* actuators (s&a)
- Each with a *periodicity* of freshness (sensors) / readiness (acts.)
  - action every *t*
- Each requires
  - computation every *t*
  - communication between s → a



# Real-Time/Cyber-Physical Systems

Embedded systems + **deadlines**

- How do we *think about/abstract time* in our computation?
- Task model:
  - Observation – s&a computation is *periodic*
  - Some computation every *t* … *how much computation?*



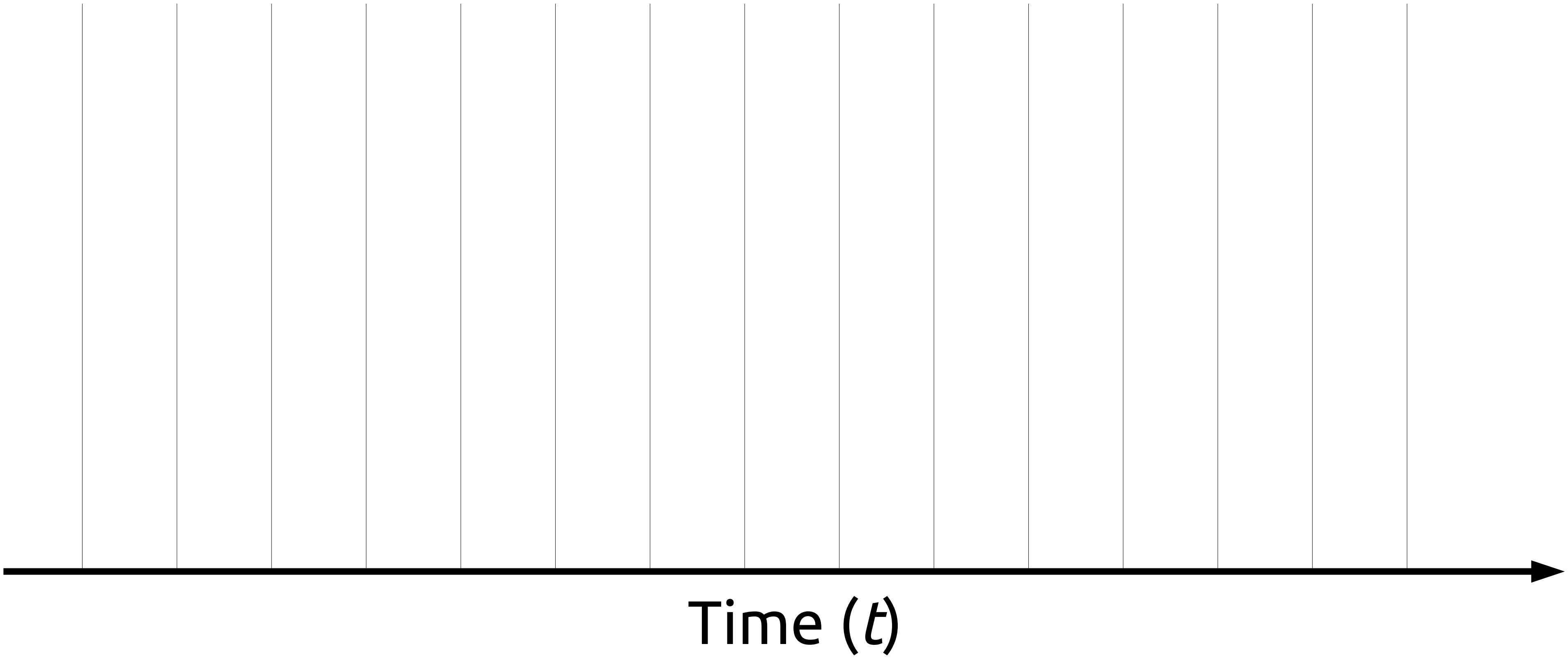
# Periodic Task Model

Task ($\tau_i$)– abstraction of computation, each has a

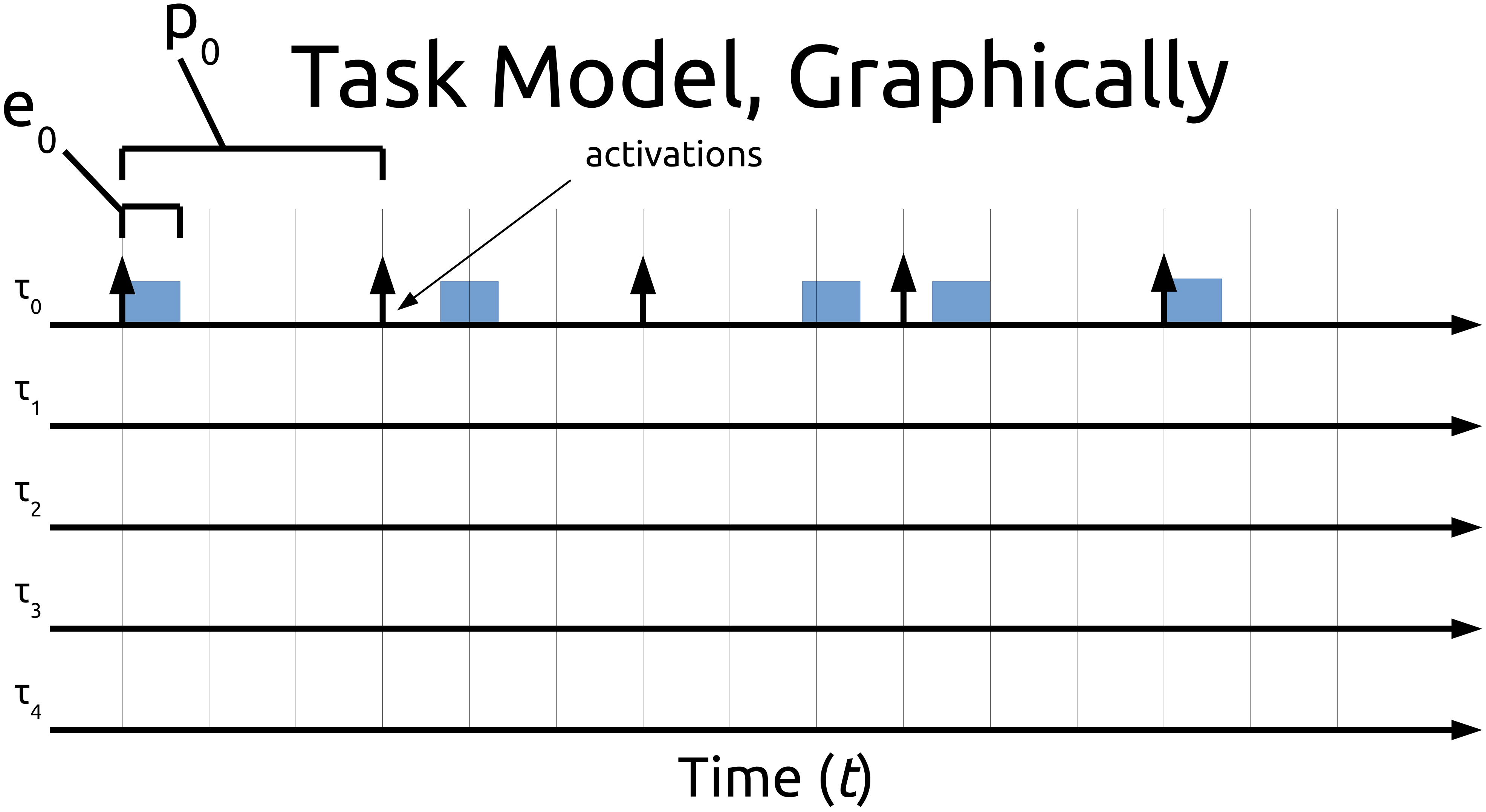
- **Period** ($p_i$) – period, span of repetitive exec.
  A task *activates* (wakes up) every $p_i$ time units
- **WCET** ($e_i$) – worst-case execution time
  Every activation, task $\tau_i$ executes for max $e_i$
- **Deadline** of $\tau_i$ is $p_i$ after activation

System is $T = \{\tau_0, \tau_1, \ldots, \tau_N\}$

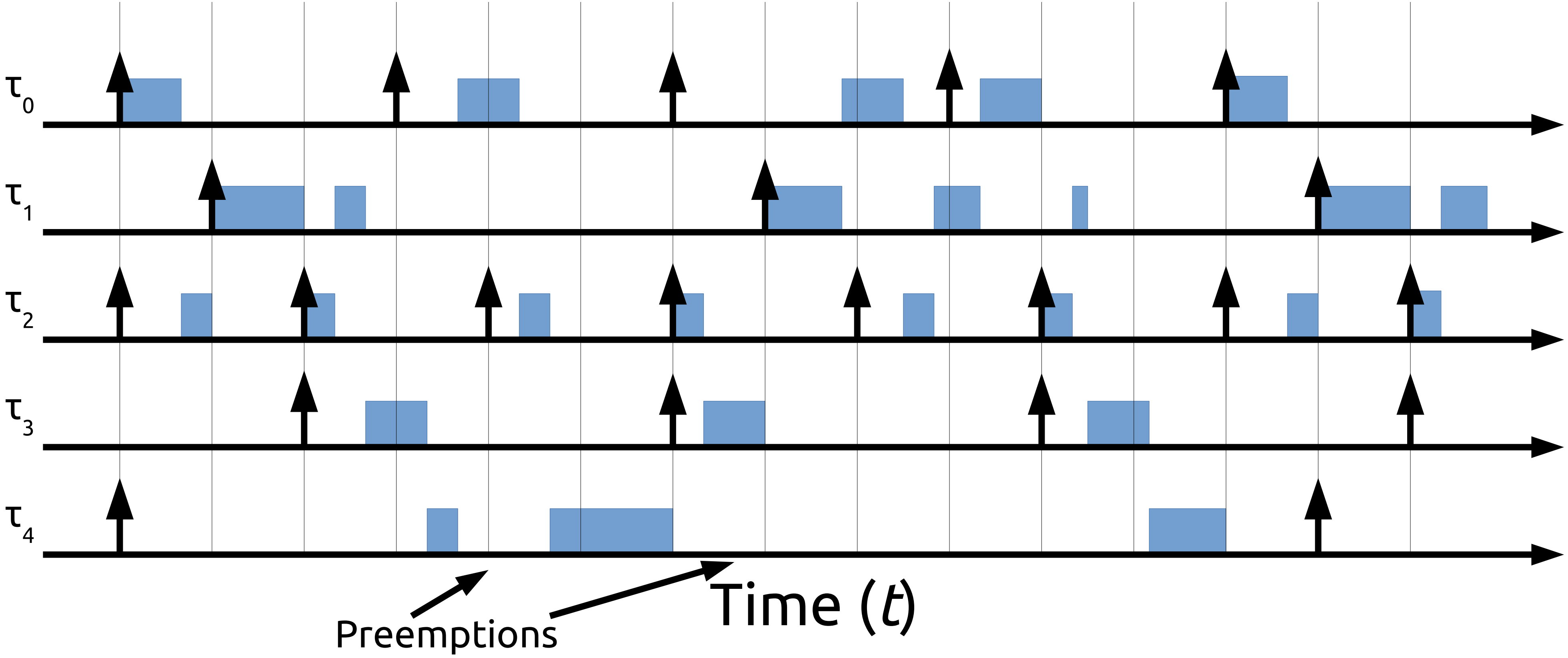
# Task Model, Graphically



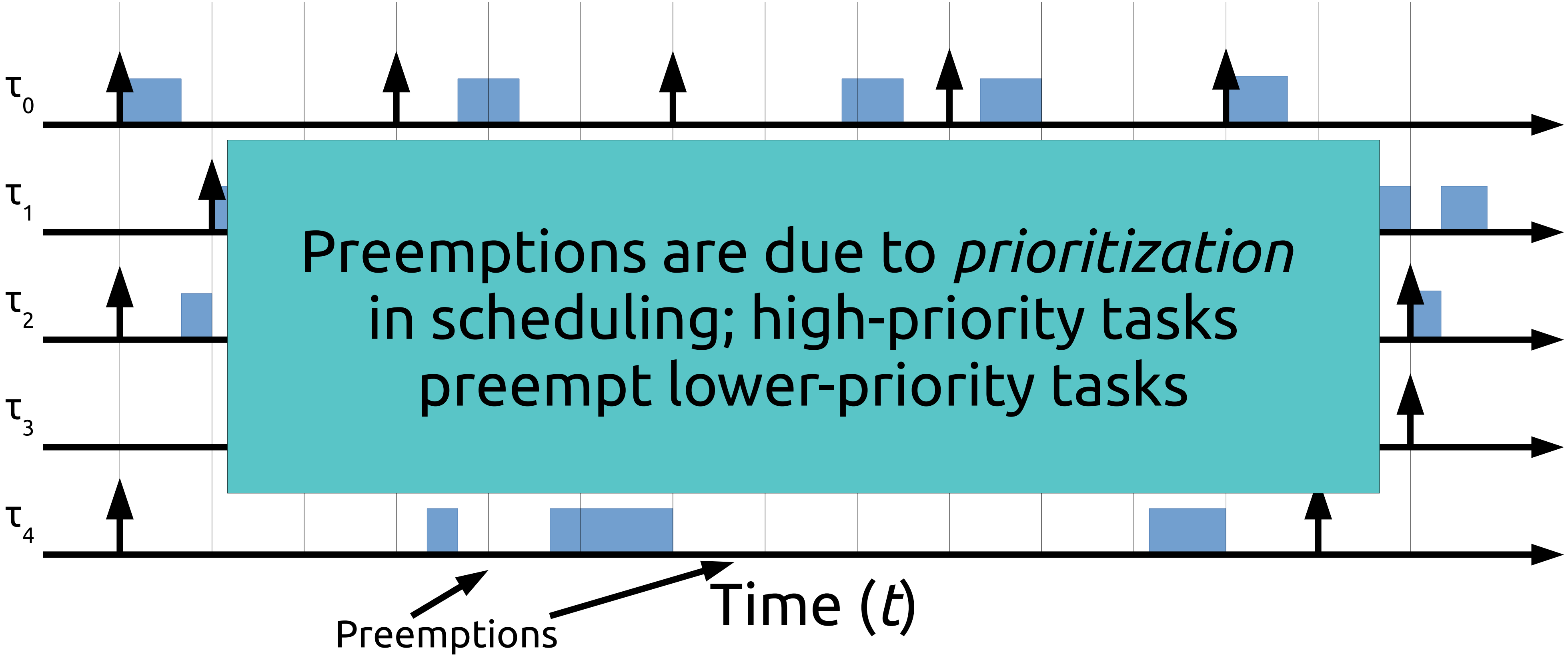
# Task Model, Graphically



# Task Model, Graphically



# Task Model, Graphically




Preemptions are due to *prioritization* in scheduling; high-priority tasks preempt lower-priority tasks

# Priority

- Fixed priority scheduling (FP): each task has a priority ($\sigma_i$)
  - highest priority, active task always executed
- Dynamic priority scheduling: earliest deadline first (EDF)
  - always execute task with *earliest deadline*
- We'll only discuss (FP)

# Fixed Priority Assignment

- How do we assign a priority, $\sigma_i$, to each task?
- Example:
  - $\tau_0$: $e_0 = 2$, $p_0 = 5$ with *low priority*
  - $\tau_1$: $e_1 = 4$, $p_1 = 10$ with *high priority*
- Note: only 8/10 time units spent on execution
- *Will we meet all deadlines?*

# Fixed Priority Assignment

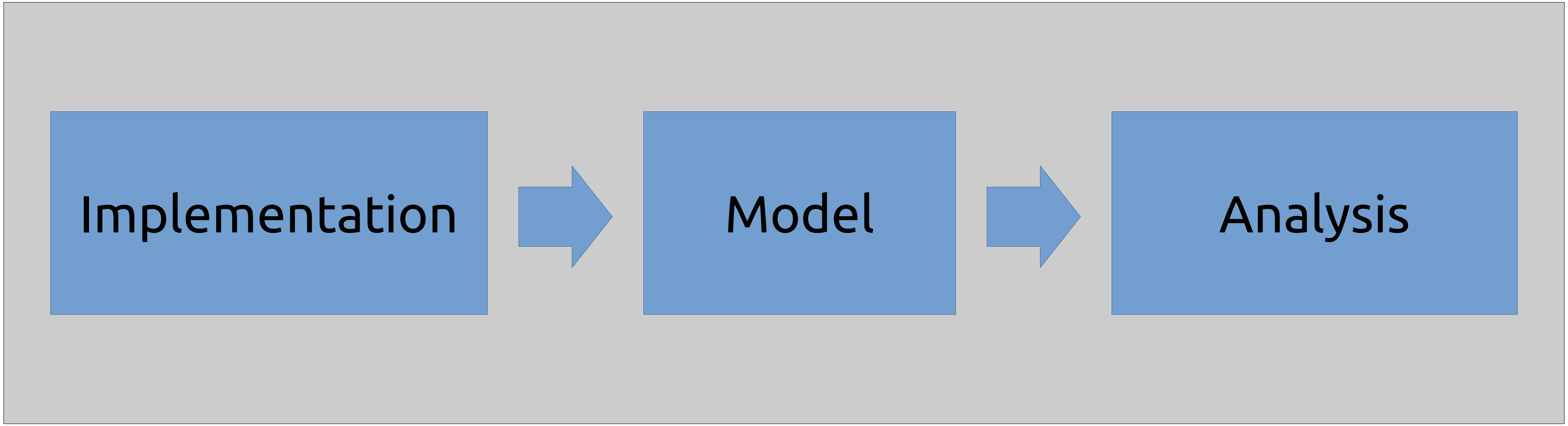
- How do we assign a priority, $\sigma_i$, to each task?
- *Ideas?*

# Fixed Priority Assignment

- How do we assign a priority, $\sigma_i$, to each task?
- One option: “Rate-Monotonic” assignment
  - Lower $p_i$ = higher priority
  - Faster rate = higher priority
  - Infrequent execution = lower priority

# Why task models?

- A model lets you talk about a *specifically-specified idea* rather than *messy implementations*
- *Goal*: real-time systems
  - **Implementation** → policies to instantiate a model
  - **Model** → mathematical description of impl.
  - **Analysis** → derive properties from model




- *Goal*: real-time systems
  - **Implementation** → policies to instantiate a model
  - **Model** → mathematical description of impl.
  - **Analysis** → derive properties from model

# Analysis: Schedulability

- Simple question:
  *Can the system meet every task's deadlines?*
- A mathematical analysis
- …using the properties of the task model
- …that describes the implementation
- …to make sure the RT system controls the world *predictably*

# Analysis: Response-Time Analysis (RTA)

Assumes Fixed-Priority Scheduling

- What is the maximum bound to complete exec?
- Intuition: Over a window of time (*w*), calculate the *interference* ($if_i(w)$) from higher-prio tasks

$if_i(p_i) + e_i > p_i$
→ cannot meet deadline
→ system not schedulable

# How would you calculate interference?

First scenario: Interference for $\tau_0$

- $\tau_0$: $e_0 = 2$, $p_0 = 5$ with *low priority*
- $\tau_1$: $e_1 = 4$, $p_1 = 10$ with *high priority*

# How would you calculate interference?

Second scenario: Interference for $\tau_2$

- $\tau_0$: $e_0 = 1$, $p_0 = 3$ with *high priority*
- $\tau_1$: $e_1 = 3$, $p_1 = 6$ with *middle priority*
- $\tau_2$: $e_2 = 1$, $p_2 = 9$ with *low priority*

# Response-Time Analysis

- Recall, we want to know if: $\boxed{\mathit{if}_i(p_i) + e_i > p_i}$
- $RTA_i(w)$ = execution time and inteference

$$RTA_i(w) = e_i + \sum_{HP(i)\ni j} \lceil RTA_i(w)/p_j \rceil e_j$$

HP(i) → all tasks with higher priority than $\tau_i$

# Response-Time Analysis

$$RTA_i(w) = e_i + \sum_{HP(i)\ni j} \lceil RTA_i(w')/p_j \rceil e_j$$

HP(i) → all tasks with higher priority than $\tau_i$

Algorithm:

```
p = e_i
r = RTA_i(p)          // start by seeing RTA over the exec time
while (p != r):       // have we converged?
    p = r
    r = RTA_i(r)      // compute the next span
```

# Interference for $\tau_2$

- $\tau_0$: $e_0 = 1$, $p_0 = 3$ with *high priority*
- $\tau_1$: $e_1 = 3$, $p_1 = 6$ with *middle prio*
- $\tau_2$: $e_2 = 1$, $p_2 = 9$ with *low priority*

$$RTA_i(w) = e_i + \sum_{HP(i)\ni j} \lceil RTA_i(w')/p_j \rceil e_j$$

$$RTA_2(1) = 1 + \lceil 1/6 \rceil 3 + \lceil 1/3 \rceil 1 = 1 + 3 + 1 = 5$$
$$RTA_2(5) = 1 + \lceil 5/6 \rceil 3 + \lceil 5/3 \rceil 1 = 1 + 3 + 2 = 6$$
$$RTA_2(6) = 1 + \lceil 6/6 \rceil 3 + \lceil 6/3 \rceil 1 = 1 + 3 + 2 = 6$$

# Interference for $\tau_2$

- $\tau_0$: $e_0 = 1$, $p_0 = 3$ with *high priority*
- $\tau_1$: $e_1 = 3$, $p_1 = 6$ with *middle prio*
- $\tau_2$: $e_2 = 1$, $p_2 = 9$ with *low priority*

$$RTA_i(w) = e_i + \sum_{HP(i)\ni j} \lceil RTA_i(w')/p_j \rceil e_j$$

$$RTA_2(1) = 1 + \lceil 1/6 \rceil 3 + \lceil 1/3 \rceil 1 = 1 + 3 + 1 = 5$$

$$RTA_2(5) = 1 + \lceil 5/6 \rceil 3 + \lceil 5/3 \rceil 1 = 1 + 3 + 2 = 6$$

$$RTA_2(6) = 1 + \lceil 6/6 \rceil 3 + \lceil 6/3 \rceil 1 = 1 + 3 + 2 = 6$$

# Interference for $\tau_2$

- $\tau_0$: $e_0 = 1$, $p_0 = 3$ with *high priority*
- $\tau_1$: $e_1 = 3$, $p_1 = 6$ with *middle prio*
- $\tau_2$: $e_2 = 2$, $p_2 = 9$ with *low priority*

$$RTA_i(w) = e_i + \sum_{HP(i)\ni j} \lceil RTA_i(w')/p_j \rceil e_j$$

$$RTA_2(2) = 2 + \lceil 2/6 \rceil 3 + \lceil 2/3 \rceil 1 = 2 + 3 + 1 = 6$$

$$RTA_2(6) = 2 + \lceil 6/6 \rceil 3 + \lceil 6/3 \rceil 1 = 2 + 3 + 2 = 7$$

$$RTA_2(7) = 2 + \lceil 7/6 \rceil 3 + \lceil 7/3 \rceil 1 = 2 + 6 + 3 = 11$$

$$11\ (RTA_2) > 9\ (p_2)$$

# Interference for $\tau_2$

- $\tau_0$: $e_0 = 1$, $p_0 = 3$ with *high priority*
- $\tau_1$: $e_1 = 3$, $p_1 = 6$ with *middle prio*
- $\tau_2$: $e_2 = 2$, $p_2 = 9$ with *low priority*

$$RTA_i(w) = e_i + \sum_{HP(i)\ni j} \lceil RTA_i(w')/p_j \rceil e_j$$

$$RTA_2(2) = 2 + \lceil 2/6 \rceil 3 + \lceil 2/3 \rceil 1 = 2 + 3 + 1 = 6$$

$$RTA_2(6) = 2 + \lceil 6/6 \rceil 3 + \lceil 6/3 \rceil 1 = 2 + 3 + 2 = 7$$

$$RTA_2(7) = 2 + \lceil 7/6 \rceil 3 + \lceil 7/3 \rceil 1 = 2 + 6 + 3 = 11$$

$$11\ (RTA_2) > 9\ (p_2)$$

# Backing up: *What have we done?*

1) Created a task model

2) Backed up by an implementation (FP sched)

3) Analyzed the task model
   - RTA: pass for all tasks → system is *schedulable*

# Additional Concerns

- “Shared resources” – locks around shared data-structures between tasks
  1) Low-priority task takes lock
  2) Preemption, switch to high-priority task
  3) High-priority task attempts to take lock

  → *high-prio task suffers low-prio interference*
- Task switches aren’t free; how analyze?
  → *all tasks suffer from dispatch latency inteference*

# Additional Concerns

- "Shared resources" – locks around shared data-structures between tasks*
- Task switches aren't free

$$RTA_i(w) = e_i + e_{lock} + \sum_{HP(i)\ni j} \lceil RTA_i(w')/p_j \rceil (e_j + 2(e_{swtch}))$$

* Assumes predictable resource sharing protocol (e.g. Priority Inheritance)

# Multicore

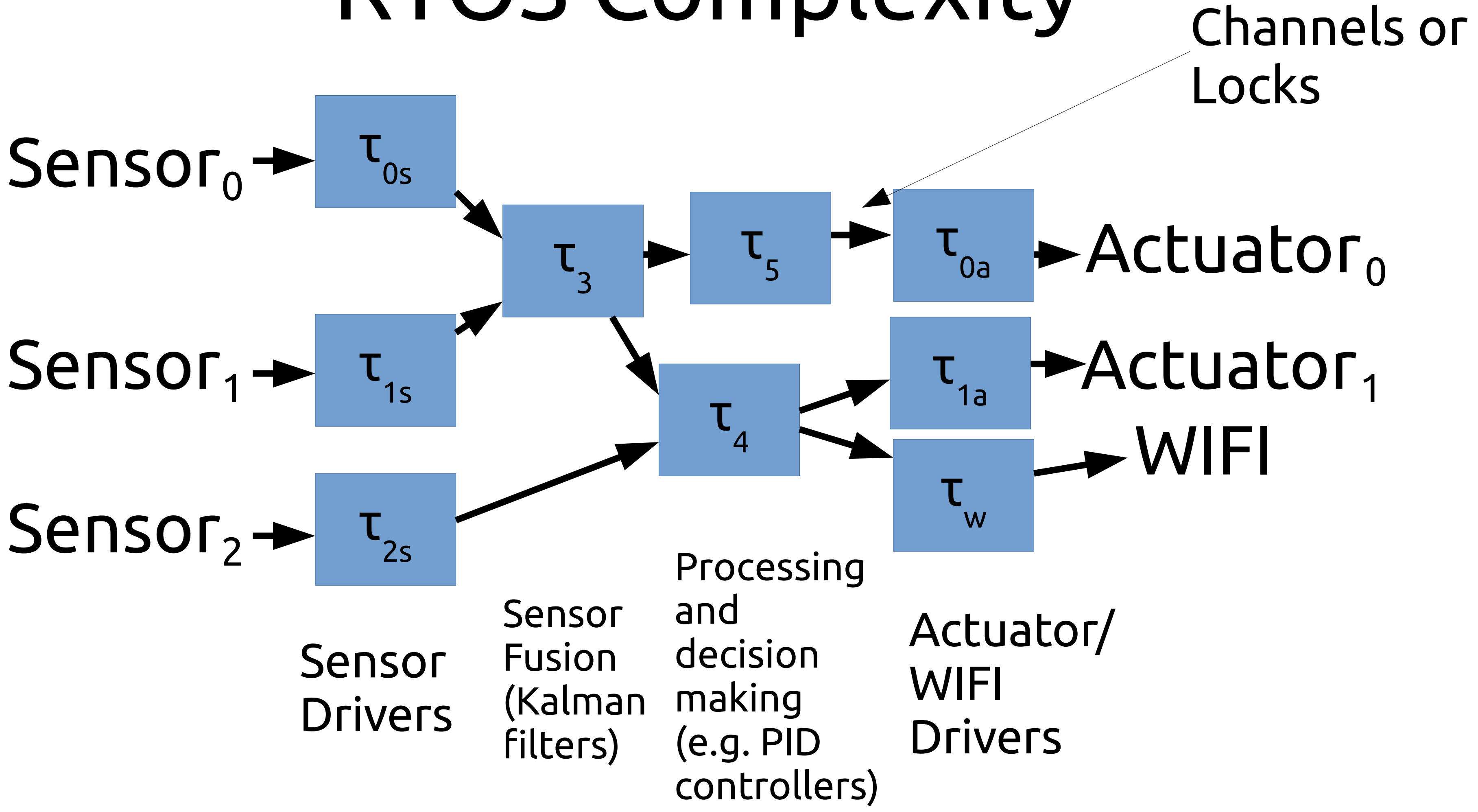
- Partitioned: Each core has its own set of tasks, and only schedules those

  → Just do an RTA per core!

- Global: All cores share the same runqueue of tasks
  - Not common

  → Need another analysis

# RTOS Complexity



# RTOS Complexity